САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Павлов Андрей Германович

Выпускная квалификационная работа бакалавра

Разработка программ для исследования характеристик веб-пространства крупной организации.

Направление 010400.62

Прикладная математика и информатика

Научный руководитель,
доктор технических наук,
доцент
Печников А.А.

Санкт-Петербург

2017

Содержание

Введение

В настоящее время задача исследования веб-пространства организаций является актуальной в связи со стремительным развитием сети интернет и ресурсов, представленных в ней. Эти исследования помогают определить, насколько организация следит за тенденцией развития своих сайтов и предоставляет результаты своей деятельности.

Веб-сайт – совокупность html-страниц и веб-документов, связанных внутренними гиперссылками [6] и обладающих единством содержания, идентифицируемая в Вебе по уникальному доменному имени.

Определим внутренние гиперссылки, как гиперссылки, которые ссылаются на html-страницы заданного веб-пространства, при этом URL-источник является также html-страницей из этого веб-пространства.

Веб-пространство организации – это множество, состоящее из веб-сайтов организации, которые связаны между собой гиперссылками. У веб-пространства всегда можно выделить его “головной сайт”, официальный сайт организации.

В данной работе будут рассмотрены веб-пространства нескольких университетов России (к примеру, Санкт-Петербургского государственного университета, Московского государственного университета), нескольких научных институтов (к примеру, Российской академии наук, Института вычислительных технологий СО РАН) и нескольких крупных организаций России (к примеру, ПАО “Газпром”, Роснефть).

Уровень веб-страницы определим следующим образом: начальная страница, передаваемая краулеру, определяемая по уникальному доменному имени, имеет уровень 0. Уровень любой другой страницы – это минимальное количество внутренних гиперссылок, ведущих от начальной страницы к данной.

Для описания веб-пространства можно использовать веб-граф. В общем случае веб-граф – это ориентированный граф, вершинами которого

являются html-страницы, ребра – гиперссылки связывающие данные вершины. В данной работе веб-граф будет представлен в виде списочной структуры, состоящей из пар сайтов (сайт1, сайт2) организации и количества дуг между ними (количество всех гиперссылок ссылающихся с сайта1 на сайт2).

Для того, чтобы построить веб-граф сайта, необходимо получить сведения о его структуре: html-страницы и гиперссылки связывающие их. В частности в данной работе необходимо получить URL-адреса веб-сайтов веб-пространства и внутренние гиперссылки между ними. Для сбора данной информации необходима программа-краулер. Краулер или же поисковой робот – программа, предназначенная для перебора страниц сети Интернет с целью сбора и/или занесения определённой информации в некую базу знаний. С общими принципами разработки краулера можно ознакомиться в работе [4].

Постановка задачи

Основная цель работы заключается в исследовании характеристик и сравнении веб-пространств нескольких крупных организаций. Для достижения данной цели были сформулированы следующие задачи, которые необходимо реализовать в работе:

1. Разработка программы-краулера, которая даёт на выходе:
    a. список всех сайтов, имеющих доменные имена, являющиеся поддоменами любого уровня доменного имени официального сайта организации.
    b. список всех гиперссылок, связывающих сайты, входящие в список из п.1.a.
2. Построение веб-графа на основе данных, полученных программой-краулером.
3. Разработка программ вычисления основных характеристик для построенного веб-графа.

4. Исследование 15 веб-пространств крупных организаций на основе построенного веб-графа.
5. Сравнение полученных характеристик веб-графов.

Глава 1. Теоретическое описание работы

1.1 Краулер

1.1.1 Описание краулера

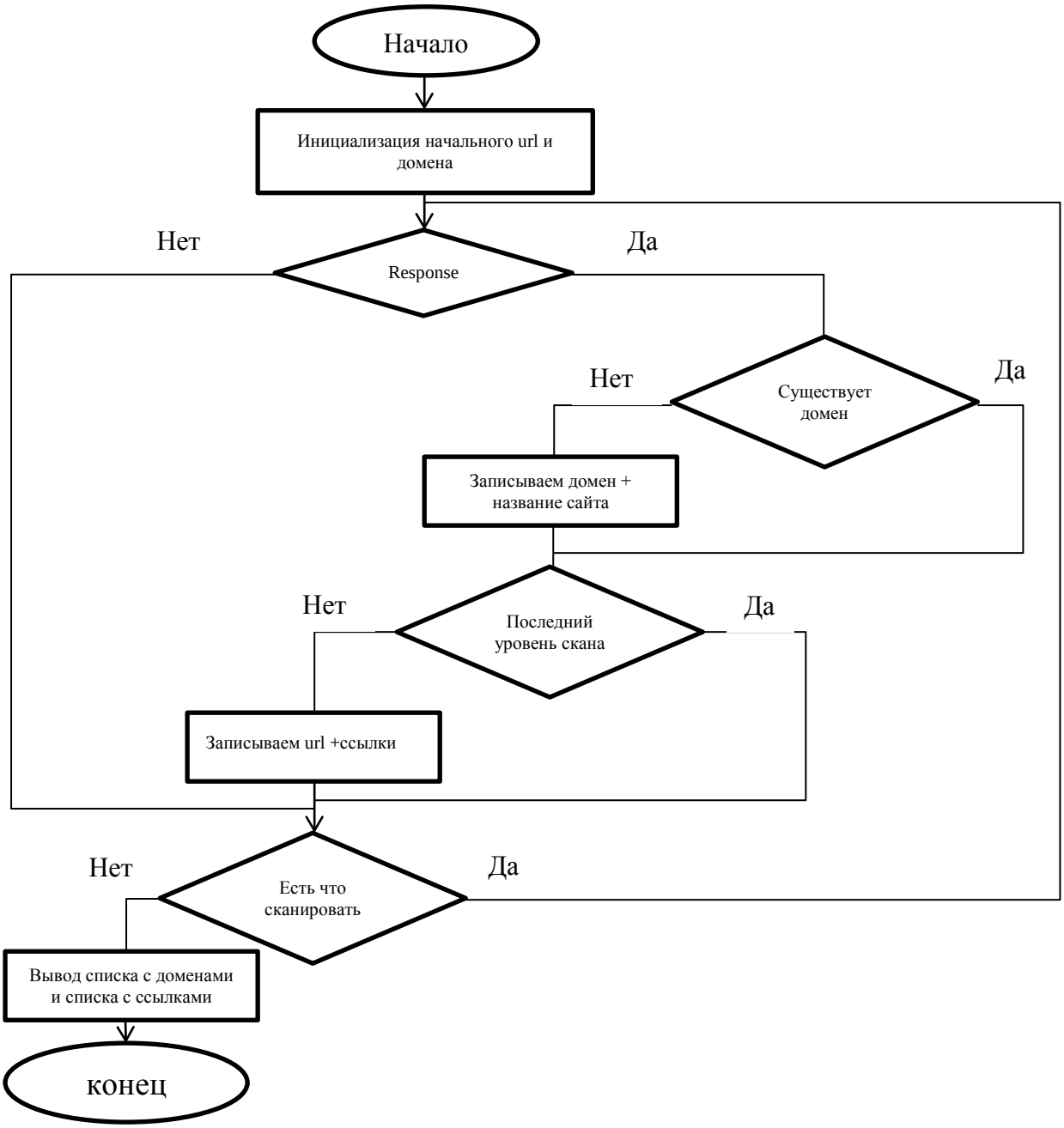
В данном параграфе описаны основные требования к разрабатываемому краулеру.

1. В качестве исходных данных подаётся доменное имя головного сайта исследуемого веб-пространства крупной организации (фактически – адрес его начальной страницы) и максимальная глубина сканирования каждого сайта веб-пространства.
2. Обход каждого сайта, начиная с главной заданной страницы, осуществляется “в ширину” по внутренним гиперссылкам.
3. Объекты сканирования – html-страницы. Гиперссылки, указывающие на файлы с расширениями rar, docx, 7z и тому подобное, и гиперссылки типа “mailto:” не рассматриваются.
4. Гиперссылки извлекаются из html-страницы на которой осуществляется сканирование из тегов <a> параметра <href>, доменное имя которых является поддоменом любого уровня доменного имени главной страницы.
5. Для гиперссылки сервер должен выдавать ответ с кодом состояния HTTP равным 200 (OK – запрос успешен) [1].
6. Страницы имеющие идентичное содержимое, но различный URL (к примеру http://spbu.ru и http://spbu.ru/) считаются одной и той же страницей и сканируются только один раз.
7. Сканирование осуществляется до тех пор, пока не будет достигнута заданная глубина сканирования, либо список страниц которые необходимо посетить будет пуст (не считая начальной итерации с

начальной страницы).

8. В качестве результата выдаётся два файла: первый содержит список всех найденных сайтов, доменное имя которых является поддоменом любого уровня доменного имени главной страницы и официальное название сайта; второй содержит список всех полученных гиперссылок, связывающих сайты из первого файла.

1.1.2 Блок-схема программы-краулера



1.2 Веб-граф

Веб-граф – это множество $G(V,E)$ состоящее из html-страниц и/или документов, являющимися вершинами V веб-графа G, и гиперссылок E, связывающих элементы из множества V.

Списочная структура веб-графа представляет из себя таблицу, столбцами которой являются пары вершин веб-графа, соответствующие дугам из E и численная мера, равная количеству гиперссылок ссылающихся с сайта столбца1 на сайт из столбца2.

При помощи веб-графа упрощается задача исследования некоторых характеристик веб-пространства.

1.3 PageRank

PageRank – это числовая величина, характеризующая “важность” веб-страницы [5]. Чем больше ссылок на страницу, тем она становится “важнее”. Кроме того, “вес” страницы A определяется весом ссылки, передаваемой от страницы B. Таким образом, PageRank – это метод вычисления веса страницы путём подсчёта важности ссылок на неё.

Очень важным является то, какое значение статистического веса имеет та страница, с которой ведёт ссылка на документ, так как по ссылке передаётся часть значения PageRank страницы “источника”.

Кроме того, если со страницы “источника” выходит несколько ссылок, одна из которых ссылается на наш документ, то передаваемый статический вес будет поделён между всеми страницами “приёмниками”.

PageRank влияет на ранжирование сайта, и если на продвигаемый сайт будет ссылаться множество ссылок со страниц с высоким PageRank, то от этого PageRank сайта вырастет.

Ниже представлена формула по которой рассчитывается PageRank страниц:

$$PR(A) = \ (1-d) + d\left(\frac{PR(P_1)}{C(P_1)} + \cdots + \frac{PR(P_n)}{C(P_n)}\right),$$

где $PR(A)$ – это вес PageRank веб-страницы A,

d – это демпфирующий коэффициент или коэффициент затухания. Он отражает какую долю веса может передать страница. Обычно его устанавливают равным 0,85,

$PR(P_1)$ – это вес PageRank веб-страницы, указывающей на страницу A,

$C(P_1)$ – это число ссылок с веб-страницы P_1.

Алгоритм реализованный в данной работе:

1. Изначально веса всех страниц нашего веб-пространства приравниваются $1/n$, где n количество вершин построенного веб-графа.
2. Для каждой веб-страницы пересчитывается её PageRank согласно описанной выше формуле.
3. П.2 выполняется до тех пор, пока $\sum_{i=1}^{n} PR(P_i)^2$ на данной итерации отличается меньше чем на $\varepsilon = 10^{-5}$ от $\sum_{i=1}^{n} PR(P_i)^2$ на прошлой итерации.

1.4 Компонента сильной связности

Компонентой сильной связности называется такое максимальное по включению подмножество, что из любой вершины данного подмножества существует направленный путь в любую другую вершину.

Для поиска компонент сильной связности был реализован алгоритм Косарайю [3]. Идея алгоритма в том, что сначала выполняется обход в “глубину” и вычисляется вектор обратного порядка обхода. Затем используется обращение этого вектора, чтобы выполнить обход в “глубину” на обращении исходного графа в котором рёбра инвертированы. Полученные при данном обходе списки являются компонентами сильной связности.

Для реализации алгоритма оперируются 3 списка:

1. *Used* – это список куда помещаются посещенные вершины на прошлых итерациях.

2. *Order* – это список, следуя которому будет осуществлён второй обход в “глубину” на обращении исходного графа.
3. *Scc* – это список куда помещаются уже готовые компоненты сильной связности.

Алгоритм реализованный в данной работе:

1. Начинается обход в “глубину” для начальной вершины *v* нашего веб-графа и она помечается как пройденная, то есть вершина *v* помещается в список *order*.
2. Выбирается вершина *v1* из списка вершин построенного веб-графа, если *v1* не содержится в *order*, то переходим к п.1. и рекурсивно вызывается работа данного алгоритма (п.1 – п.3).
3. Записывается вершина *v* в список *order* и осуществляется переход к новой вершине из списка вершин веб-графа.
4. Создаётся новый чистый список *used*.
5. Начинается обход в “глубину” на обращении исходного графа (рёбра инвертированы), вершина выбирается последней из списка *order*, которая ещё не была посещена.
6. Посещённая вершина записывается в список *order*.
7. Если вершина *v* не содержится в списке *order* , то рекурсивно вызывается работа данного алгоритма (п.5 – п.7).
8. Как только была получена вершина, которая содержится в списке *order*, все пройденные вершины данного обхода в глубину записываются в список *Scc*.

1.5 Клика в орграфе

Кликой ориентированного графа называется подмножество его вершин такое, что для любых его двух вершин *s*, *t* существует дуга из *s* в *t* и наоборот.

Максимальная клика – это клика, которая не может быть дополнена путём включения дополнительных смежных вершин, то есть нет клики

большего размера, включающей все вершины данной клики.

Для поиска всех клик в ориентированном графе был реализован алгоритм Брона – Кербоша [2], с условием что наш граф ориентированный. Идея в том, что всякая клика считается его максимальном по включение полным подграфом. На каждой итерации алгоритм пытается дополнить уже построенный полный подграф вершинами из списка *candidates*. Высокая скорость достигается за счёт отсечения вершин, которые были использованы для увеличения полного подграфа. Такие вершины помещаются в список *not*.

Всего используется 3 списка:

1. *Compsub* – это список, содержащий полный подграф на каждом шаге рекурсии.
2. *Candidates* – это список вершин, которые могу увеличить *compsub*.
3. *Not* – это список, содержащий вершины которые уже использовались для расширения *compsub* на предыдущих итерациях алгоритма.

Ниже представлен алгоритм:

1. Передаются исходные данные: список *candidates* состоящий из всех вершин веб-графа, пустые списки *not*, *compsub*.
2. Выбирается вершина *v* из списка *candidates* и добавляется в список *compsub*.
3. Формируются новые списки *new_candidates* и *new_not* путём удаления вершин из *candidates* и *not*, не соединённых с вершиной *v* соответственно.
4. Если *new_candidates*, *new_not* пусты и размер списка *compsub* больше заданной величины, то выводится *compsub*.
5. Если *new_candidates*, *new_not* не пусты, то переходим к п.1 с новыми списками и рекурсивно вызываем работу данного алгоритма (п.1 – п.7).
6. Удаляем вершину *v* из *compsub* и *candidates*, добавляем

вершину в *not*.

7. Если *candidates* не пуст и *not* не содержит вершины, соединённой со всеми вершинами из списка *candidates*, то переходим к п.1.

Глава 2. Результаты работы

2.1 Практическая реализация

2.1.1 Инструменты

Для реализации краулера и программ, что вычисляют характеристики веб-графа, был использован язык программирования Java в интегрированной среде разработки программ Intellij Idea [9].

Для парсинга html-страниц был выбран парсер Jsoup [8], который содержит все необходимые функции для реализации программы.

Для записи выходных данных всех разработанных программ использовалась Javenue.csv [11], которая позволяет читать и записывать данные в файл формата csv.

При визуализации графа был использован JGraph [10], довольно лёгкая библиотека для того чтобы быстро построить и вывести на экран нужный граф.

2.1.2 Работа краулера

Рассмотрим детально работу программы-краулера на примере официального сайта Санкт-Петербургского государственного университета (СПбГУ): http://spbu.ru.

При начальном URL: http://spbu.ru и заданной глубине сканирования 4 программой-краулер было посещено 24590 страниц. Первый список из Постановка задачи п.1.а насчитывает 151 веб-сайт, ниже представлена таблица с некоторыми значениями списка 1.

Доменное имя веб-сайта	**Официальное название веб-сайта**
spbu.ru	СПбГУ
english.spbu.ru	SPBU - Saint Petersburg University

chinese.spbu.ru	SPBU -圣彼得堡国立大学
dspace.spbu.ru	DSpace at Saint Petersburg State University: Главная страница
forum.spbu.ru	Общественное обсуждение
events.spbu.ru	Управление по организации публичных мероприятий и сотрудничества с партнерами СПбГУ - Главная
nauka.spbu.ru	Наука в СПбГУ
abiturient.spbu.ru	Приемная комиссия - abiturient.spbu.ru
alumni.spbu.ru	Ассоциация выпускников СПбГУ
fund.spbu.ru	Эндаумент-фонд СПбГУ - Главная
pobeda.spbu.ru	70-летие Великой Победы - Сражающийся Университет
pr.spbu.ru	Новости - Фирменный стиль СПбГУ
gsom.spbu.ru	ВШМ СПбГУ - бизнес-школа №1 в России
earth.spbu.ru	Институт наук о Земле
jf.spbu.ru	ГЛАВНАЯ :: Институт «Высшая школа журналистики и массовых коммуникаций» СПбГУ
history.spbu.ru	Институт
philosophy.spbu.ru	ИФ СПбГУ — Институт философии СПБГУ
chem.spbu.ru	Институт - Институт химии СПбГУ
bio.spbu.ru	Новости и анонсы событий факультета
orient.spbu.ru	Главная - Восточный факультет СПбГУ
arts.spbu.ru	Факультет искусств

Таблица 1. Доменные имена и официальные названия веб-сайтов СПбГУ.

При заданном сканировании были найдены главный сайт СПбГУ, английская и китайская версии сайта СПбГУ, все факультеты СПбГУ (факультет прикладной математики и процессов управления (apmath.spbu.ru), факультет социологии (soc.spbu.ru), юридический факультет (law.spbu.ru)), сайт приёмной комиссии, расписание сайта СПбГУ и другие.

Больше всего доменных имён, являющихся поддоменом любого уровня

домена главного сайта, при идентичной заданной глубине сканирования, было найдено при сборе данных сайта Московского государственного университета (МГУ). Было найден 291 веб-сайт, из которых главный сайт МГУ, факультеты сайта МГУ, новостные сайты, лаборатории, сайты приемной комиссии, олимпиад и другие.

Из 291 веб-сайта, 142 веб-сайта являются поддоменом третьего уровня домена msu.ru главного сайта МГУ. На примере официального сайта СПбГУ из 151 веб-сайта только 25 являются поддоменами третьего уровня домена spbu.ru.

Ещё одно интересное наблюдение, сравнивая веб-пространства СПбГУ и МГУ: для примера официального сайта СПбГУ spbu.ru было просканировано 24590 html-страниц, в то время как для примера официального сайта МГУ msu.ru просканировано 14840 html-страниц. При этих показателях были получены следующие данные веб-пространств:

1. СПбГУ – 151 веб-сайт и 99930 гиперссылок, связывающих найденные 151 веб-сайт.
2. МГУ – 291 веб-сайт и 80154 гиперссылок, связывающих найденные 291 веб-сайт.

Из этого можно сделать заключение, что в среднем с html-страниц веб-пространства МГУ исходит больше внутренних гиперссылок, чем с html-страниц веб-пространства СПбГУ.

Меньше всего доменных имён с помощью реализованной программы-краулера было получено для веб-пространств “ООО Пивоваренная компания Балтика”, а именно 3 (сайт пивоваренной компании Балтика, английская версия сайта и официальный сайт) и “Института космических исследований”, а именно 6 (среди них главный сайт, веб-сайт “О нас”, сайт “MTD” и другие).

Для примера spbu.ru (официального сайта СПбГУ) программа-краулер на выходе выдала список, который насчитывает 99930 гиперссылок соединяющих веб-сайты из таблицы 1, ниже представлена таблица с

некоторыми результатами.

URL-источник	**URL-премник**
http://spbu.ru	http://english.spbu.ru
http://spbu.ru	http://chinese.spbu.ru
http://spbu.ru	https://dspace.spbu.ru
http://nauka.spbu.ru/megagrany-spbgu	https://ias.spbu.ru
https://abiturient.spbu.ru/russkij/18-cat-rus/priem-rus.html	http://guestbook.spbu.ru
http://spbu.ru/anonsy/details/2/12639.html	http://chinese.spbu.ru
http://spbu.ru/konferentsii.html	http://events.spbu.ru/events/inicziacziya/inicziacziya-meropriyatiya.html
http://english.spbu.ru/our-university/saint-petersburg	https://dspace.spbu.ru/?locale=en
http://english.spbu.ru/our-university/job-at-spbu	http://spbu.ru/files/upload/staff/AcademicStaffingRegulations.pdf
http://english.spbu.ru/our-university/university-a-fresh-start	http://spbu.ru
http://chinese.spbu.ru/2016-10-15-20-33-42	http://english.spbu.ru/events
http://spbu.ru/component/banners/click/112.html	http://alumni.spbu.ru/search/?tags=%23spbu
http://jf.spbu.ru/	http://fund.spbu.ru
http://orient.spbu.ru/	https://ias.spbu.ru
http://math.spbu.ru/	http://fund.spbu.ru
http://apmath.spbu.ru/	https://my.spbu.ru
http://apmath.spbu.ru/	http://guestbook.spbu.ru
http://dent.spbu.ru/	http://iberorus.spbu.ru
http://spbu.ru/component/banners/click/120.html	http://www.econ.spbu.ru/content/news
http://edu.spbu.ru/index.php/normativnye-akty#s5_scrolltotop	http://spbu.ru/structure/documents/mm19xm7g

Таблица 2. Внутренние гиперссылки веб-пространства СПбГУ.

2.1.3 Построение веб-графа

При создании веб-графа использовались списки, полученные при сканировании краулером.

Веб-граф представлен в виде списка (домен сайта1, домен сайта2, количество гиперссылок исходящих из сайта1 на сайт2), ниже на рисунке 1 показана визуализация построенного веб-графа и показаны результаты с наибольшим количество исходящих гиперссылок.

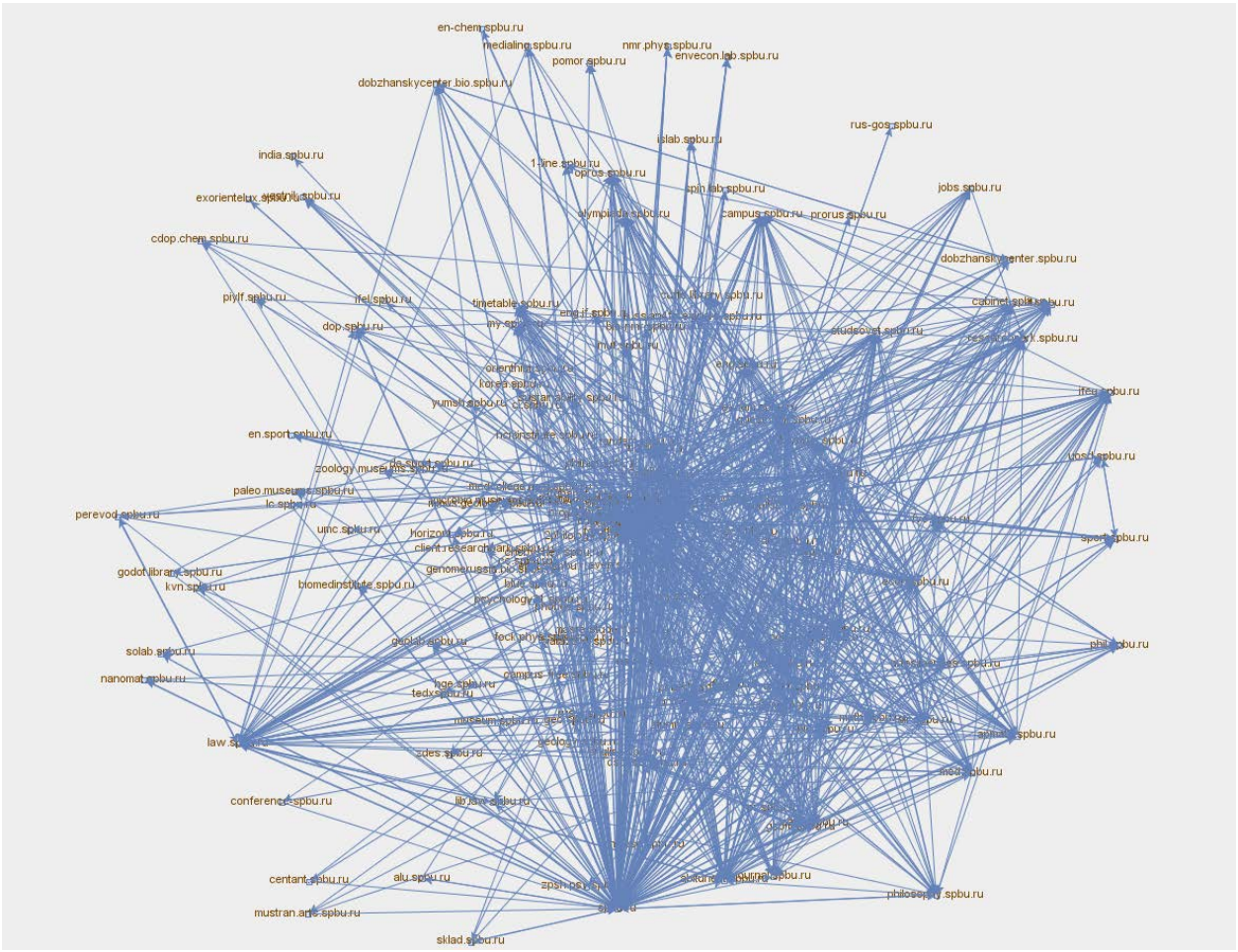


Рис. 1. Визуализация веб-графа СПбГУ.

Доменное имя источника	Доменное имя приемника	Количество дуг
guestbook.spbu.ru	spbu.ru	11664
spbu.ru	english.spbu.ru	7767
dspace.spbu.ru	spbu.ru	4432
dspace.spbu.ru	it.spbu.ru	4384
spbu.ru	chinese.spbu.ru	3883
nauka.spbu.ru	spbu.ru	2244

psy.spbu.ru	spbu.ru	2065
researchpark.spbu.ru	spbu.ru	1913
guestbook.spbu.ru	forum.spbu.ru	1811
guestbook.spbu.ru	law.spbu.ru	1808
law.spbu.ru	spbu.ru	1585
studsovet.spbu.ru	spbu.ru	1584
spbu.ru	guestbook.spbu.ru	1575
rus-gos.spbu.ru	spbu.ru	1283
psy.spbu.ru	guestbook.spbu.ru	1174
med.spbu.ru	spbu.ru	995
spbu.ru	nauka.spbu.ru	862
spbu.ru	researchpark.spbu.ru	801
chem.spbu.ru	spbu.ru	772
law.spbu.ru	guestbook.spbu.ru	737
phys.spbu.ru	abiturient.spbu.ru	724
gsom.spbu.ru	spbu.ru	657
artesliberales.spbu.ru	spbu.ru	565
dent.spbu.ru	spbu.ru	529
phys.spbu.ru	spbu.ru	527

Таблица 3. Веб-граф СПбГУ.

В принципе визуализация графа не говорит нам общим счётом ничего, поэтому для исследования будем опираться на таблицу 3 полученную в ходе работы программы-краулера.

Как видно из таблицы 3 наибольшее количество внутренних гиперссылок исходят из главного сайта СПбГУ spbu.ru, сайта виртуальной приемной комиссии СПбГУ и сайта архива открытого доступа СПбГУ.

Также в список попали несколько веб-сайтов факультетов СПбГУ (Факультет психологии law.spbu.ru, Юредический факультет law.spbu.ru), веб-сайт научной деятельности СПбГУ nauka.spbu.ru, веб-сайт студенческого совета СПбГУ studsovet.spbu.ru, веб-сайт научного парка СПбГУ researchpark.spbu.ru и другие.

2.1.4 Характеристики веб-графа

Далее при помощи реализованной программы для подсчёта характеристики PageRank веб-сайта был рассчитан PageRank всех веб-сайтов, являющихся вершинами построенного веб-графа для веб-пространства СПбГУ.

Ниже приведены первые 25 веб-сайтов с наивысшим PageRank.

Доменное имя веб-сайта	**PageRank**
spbu.ru	0.01478453715031975
publishing.spbu.ru	0.004270371762444703
orient.spbu.ru	0.0041379647234110715
sport.spbu.ru	0.003987980627049369
english.spbu.ru	0.0030760935138737248
de.sport.spbu.ru	0.003056170724327207
en.sport.spbu.ru	0.003056170724327207
fr.sport.spbu.ru	0.0028823263797461075
it.spbu.ru	0.0024758335429021904
law.spbu.ru	0.0018191063623624 37
abiturient.spbu.ru	0.0017409989014380044
phil.spbu.ru	0.001725230955905166
guestbook.spbu.ru	0.0016052289737832886
vestnik.spbu.ru	0.0015352680126627144
csr.spbu.ru	0.0014896600749494707
psy.spbu.ru	0.0014553027533353727
history.spbu.ru	0.0014507085145988464
chem.spbu.ru	0.0014369335462761311
researchpark.spbu.ru	0.001352667176746609
students.spbu.ru	0.0013218484012034052
bio.spbu.ru	0.0013174616751697738
blue.spbu.ru	0.0013119906427693183
earth.spbu.ru	0.001304767544845817
arts.spbu.ru	0.0012949996204246511
jf.spbu.ru	0.001239938768153249

Таблица 4. PageRank веб-сайтов СПбГУ.

Как видим, наивысший показатель PageRank имеет главный сайт Санкт-Петербургского государственного университета spbu.ru, что и следовало ожидать. Также в таблицу 4 попали несколько веб-сайтов факультетов СПбГУ, веб-сайт кафедры физической культуры и спорта и его английскую, немецкую и французскую версии веб-сайта.

Остальные веб-сайт имеют PageRank в диапазоне (0.0012 – 0.0009). Что означает, что в полученном веб-графе существует обособленные веб-сайты, которые не ссылаются или имеют небольшое количество исходящих ссылок на другие веб-сайты.

Интересные значения PageRank были получены при исследовании компании ПАО “Газпром”. Главный веб-сайт gazprom.ru имеет показатель PR равный 0.0029, остальные 79 веб-сайтов имеют PR примерно равный 0.002, что уже может означать, что веб-пространство компании ПАО “Газпром” хорошо связано и не имеет обособленных веб-сайтов.

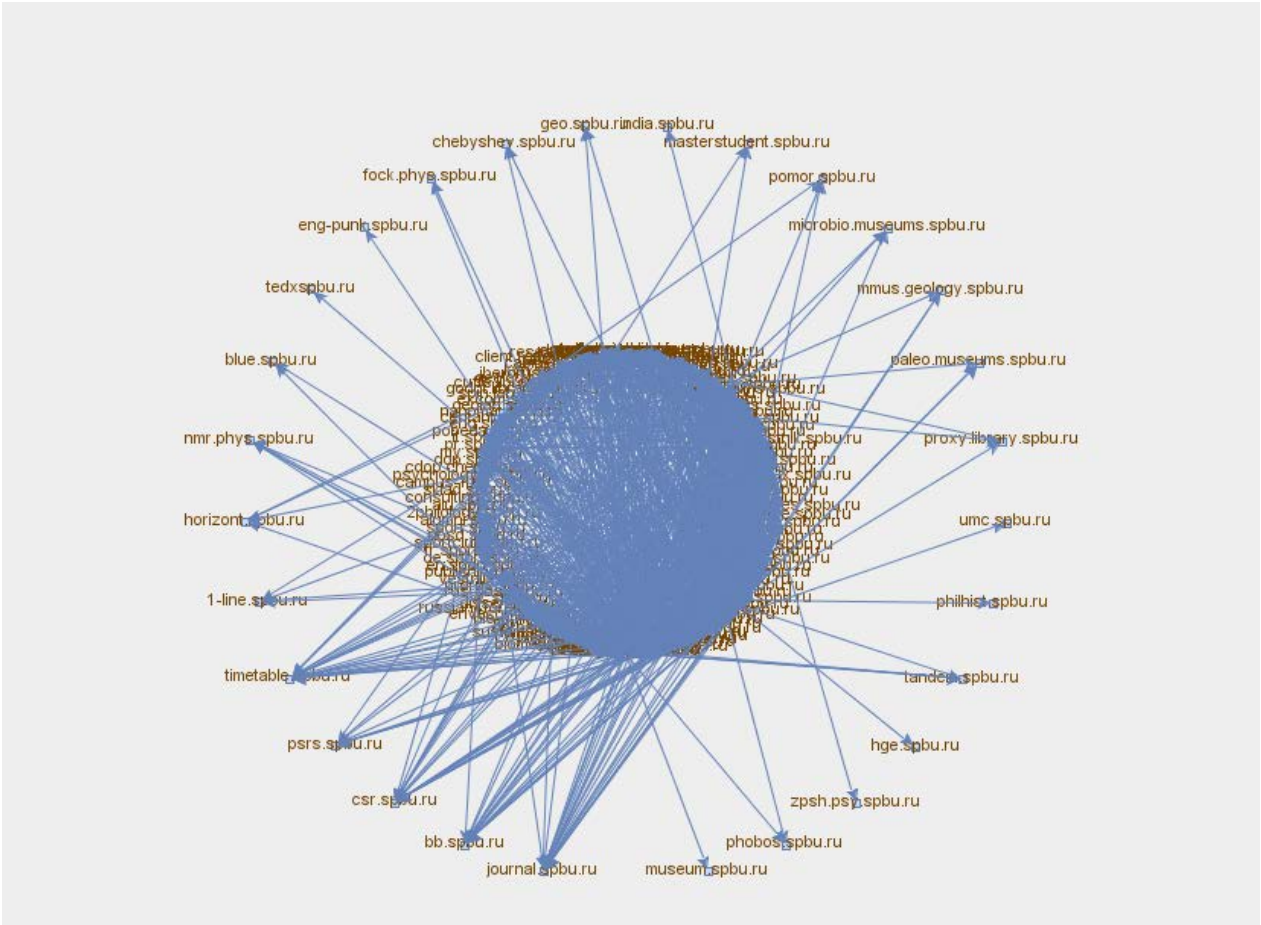


Рис. 2. Визуализация веб-графа СПбГУ с КСС.

На рисунке 2 представлены найденные компоненты сильной связности, в центре рисунка максимальная компонента, насчитывающая 123 веб-сайта, остальные 28 компонент состоят из одного веб-сайта.

Можно видеть, что обособленные 28 вершин графа не передают ни одной внутренней гиперссылки, поэтому веб-сайты, относящиеся к данным вершинам, не будут и задействованы в найденных кликах веб-пространства СПбГУ.

При реализованной программе по нахождению клик размером 3 и больше, для веб-графа СПбГУ были найдены 75 клик, максимальная клика содержит 6 веб-сайтов. При этом 74 клики содержат официальный сайт СПбГУ (spbu.ru). Единственная клика, не содержащая spbu.ru, клика – sport.spbu.ru; en.sport.spbu.ru; de.sport.spbu.ru; fr.sport.spbu.ru, состоящая из всех версий веб-сайта кафедры физической культуры и спорта.

Ниже представлены некоторые полученные клики (Строка – одна килка, количество ячеек в строке – размер клики):

spbu.ru	english.spbu.ru	chinese.spbu.ru			
spbu.ru	dspace.spbu.ru	gsom.spbu.ru			
spbu.ru	forum.spbu.ru	researchpark.spbu.ru			
spbu.ru	nauka.spbu.ru	philosophy.spbu.ru			
spbu.ru	nauka.spbu.ru	sir.spbu.ru	guestbook.spbu.ru		
spbu.ru	abiturient.spbu.ru	psy.spbu.ru	edu.spbu.ru	guestbook.spbu.ru	law.spbu.ru
spbu.ru	abiturient.spbu.ru	guestbook.spbu.ru	olympiada.spbu.ru	law.spbu.ru	
spbu.ru	fund.spbu.ru	jf.spbu.ru			
spbu.ru	gsom.spbu.ru	guestbook.spbu.ru	studsovet.spbu.ru		
spbu.ru	earth.spbu.ru	geolab.spbu.ru			
spbu.ru	jf.spbu.ru	edu.spbu.ru			
spbu.ru	chem.spbu.ru	edu.spbu.ru	guestbook.sp		

			bu.ru		
spbu.ru	bio.spbu.ru	edu.spbu.ru			
spbu.ru	orient.spbu.ru	edu.spbu.ru			
spbu.ru	psy.spbu.ru	guestbook.spbu.ru	studsovet.spbu.ru	law.spbu.ru	
spbu.ru	artesliberales.spbu.ru	library.spbu.ru			
spbu.ru	phil.spbu.ru	guestbook.spbu.ru			
spbu.ru	econ.spbu.ru	economics.vestnik.spbu.ru			
spbu.ru	students.spbu.ru	guestbook.spbu.ru	studsovet.spbu.ru		
sport.spbu.ru	fr.sport.spbu.ru	de.sport.spbu.ru	en.sport.spbu.ru		

Таблица 5. Клики веб-графа СПбГУ.

Такое большое количество клик малого размера обусловлено тем, что университет имеет много факультетов, которые вполне могут и не ссылаться друг на друга. Собственно из таблицы 5 можно видеть много клик, содержащих сайт СПбГУ, сайт одного из факультетов СПбГУ и ещё один/пару не сайтов факультета СПбГУ.

2.2 Исследование

Главный сайт веб-пространства	**Количество вершин**	**Количество дуг**	**PR главной страницы**	**Макс. клика/ Всего клик размера =>3**	**Макс. компонента сильной связности**	**Соотношение количества вершин к количеству дуг**	**Соотношение макс. клика к кол-ву вершин**	**Соотношение макс. Компонента к кол-ву вершин**
Spbu.ru	151	99930	0.0148	6/75	123	0.0015	0.0331	0.8145
Msu.ru	291	80154	0.0161	4/79	183	0.0036	0.0137	0.6288
Mipt.ru	85	26106	0.0228	3/1	60	0.0032	0.0353	0.7058
Urfu.ru	126	81777	0.0264	4/39	115	0.0015	0.0317	0.9126

Petrsu.ru	53	87964	0.0882	3/8	40	0.0006	0.0566	0.7547
Gazprom.ru	80	1278255	0.0297	73/3	77	0.00006	0.9125	0.9625
Severstal.com	27	80028	0.0318	16/5	26	0.0003	0.5929	0.9629
Rosneft.ru	69	26719	0.0205	47/22	69	0.0025	0.6811	1
Baltika.ru	3	3647	0.0503	3/1	3	0.0008	1	1
Evraz.com	10	280	0.043	5/2	8	0.0357	0.5	0.8
Kunstkamera.ru	11	479	0.273	0/0	8	0.0229	0	0.7272
Ict.nsc.ru	10	4673	0.0234	3/1	8	0.0021	0.3	0.8
Iki.rssi.ru	6	284	0.197	0/0	5	0.0211	0	0.8333
Krc.Karelia.ru	42	25641	0.1029	4/9	31	0.0016	0.0952	0.7380
Ras.ru	59	724	0.0405	3/2	25	0.0814	0.0508	0.4237

Таблица 6. Основные характеристики исследуемых веб-пространств.

В таблице 6 были отражены полученные данные по заданным веб-пространствам, также отражены основные полученные характеристики.

Из данной таблицы явно выделяется компания ПАО “Газпром” gazprom.ru, с показателями соотношений максимальной клики к количеству вершин и максимальной компоненты сильной связности к количеству вершин равны 0.9125 и 0.9625 соответственно. При этом максимальная клика и максимальная компонента сильной связности почти полностью состоят из всех вершин построенного веб-графа, который в свою очередь насчитывает 80 веб-сайтов с уникальным доменным именем и 1 278 255 гиперссылок, связывающих найденные вершины. По данным показателям можно смело утверждать, что внутреннее веб-пространство компании ПАО “Газпром” хорошо связано и практически не имеет обособленных веб-сайтов, которые несут небольшое количество информации по данному веб-пространству.

Также из данной таблицы легко выделить плохо связанные сайты и имеющие малое количество вершин и гиперссылок связывающих их, это веб-сайты:

1. Институт космических исследование iki.rssi.ru.

2. Кунсткамера (Музей антропологии и этнографии им. Петра Великого Российской Академии наук) kunstkamera.ru.
3. Компания Evraz evraz.com.

Все перечисленные выше веб-сайты содержат в построенном для них веб-графе малое кол-во вершин и рёбер. Веб-сайты содержат слабую связность при этом, два веб-сайта, кроме компании Евраз, не содержат клик размера три и больше. Из этого можно заключить, что веб-пространства данных организаций слабо развиты.

Веб-пространства пивоваренной компании “Балтика” baltika.ru и института вычислительных технологий СО РАН ict.nsc.ru также содержат малое количество веб-сайтов (3 и 10 соответственно), однако, содержат большое количество гиперссылок связывающих их. Институт вычислительных технологий СО РАН имеет много обособленных веб-сайтов, так как содержит только одну клику размера 3, и максимальная компонента сильной связности 8, при 10 веб-сайтах и 4 673 гиперссылках. А говоря о компании “Балтика”, можно сказать что веб-пространство полностью связно и веб-сайты ссылаются друг на друга.

К группе сильно связанных веб-пространств, имеющих большое количество внутренних гиперссылок, можно отнести веб-сайты:

1. Санкт-Петербургского государственного университета spbu.ru.
2. Уральского федерального университета urfu.ru.
3. Московского физико-технического университета mipt.ru.
4. Петрозаводского государственного университета petrsu.ru.
5. Вертикально-интегрированной горнодобывающей и сталелитейной компании Северсталь severstal.com.
6. Нефтегазовой компании Роснфеть rosneft.ru.
7. Карельского научного центра Российской академии наук krc.karelia.ru.

Веб-пространства компаний “Роснефеть” и “Северсталь” имеют высокий коэффициент отношения максимальной клики к количеству веб-

сайтов, 0.6811 и 0.5929 соответственно.

Веб-пространства университетов России имеют низкий коэффициент отношения максимальной клики к количеству веб-сайтов в диапазоне от 0.01 до 0.05. Это обусловлено структурой создания веб-пространства университета, так как университет подразделяется на факультеты, развивающие свои веб-сайты, имеет различные веб-сайты (к примеру, сайты научной деятельности, архивы, библиотеки и т.д.), также имеющие обособленную структуру. В следствие чего было получено обильное количество клик у построенных веб-графов, однако максимальный размер данных клик не выше 5.

К группе слабо связанных веб-пространств можно отнести Российскую академию наук ras.ru. Как и в случае структуры веб-пространства университета, веб-ресурс Российская академия наук ras.ru содержит веб-сайты научных институтов России, которые являются обособленными друг от друга и содержащие собственную структуру веб-ресура.

Веб-пространство Московского государственного университета msu.ru сложно отнести к одной из двух перечисленных групп, так как были получены спорные показатели характеристик веб-графа МГУ, требуется дополнительное исследование для решения данной проблемы.

2.3. Тестирование

Тестирование программы проводилось вручную. На вход подавались URL несуществующих страниц, некорректные URL, страницы с различными протоколами URL и др. По завершении тестирования построенной программой-краулером в данной работе были посещены страницы протокола HTTP и HTTPS [12], проигнорированы страницы URL иных протоколов, URL ведущие к скачиванию файлов, URL несуществующих страниц.

Информация со страниц которые уже посетили, но URL различен (в конце гиперссылки стоит символ “/”, или “/index.html”, или просто различные URL, что ведут к одинаковой html-странице), не записывалась повторно.

Время посещения одного веб-сайта (то есть, получение ответа от веб-сервера и работа всех методов) составляет от 0.1 до 3 секунд. Таким образом, работа программы в среднем занимает пару часов.

Выводы

В ходе данной выпускной квалификационной работы был разработан краулер для сбора информации и последующего построения веб-графа исследуемых организаций. Были разработаны программы нахождения основных характеристик веб-графа: PageRank, компонента сильной связности и клика ориентированного графа.

Были исследованы 15 веб-пространств крупных организаций из которых 5 – университеты России, 5 – крупные компании, 5 – научные институты.

Были сравнены полученные результаты, в итоге все веб-пространства удалось разделить на несколько групп: сильно/слабо связанных веб-пространств, веб-пространств со слабой структурой, веб-пространств с обильным числом обособленных веб-сайтов.

Заключение

Задачу, рассмотренную в данной работе, в дальнейшем следует исследовать более детально. К примеру, для улучшения показателей можно воспользоваться методами кластерного анализа. Кластерный анализ – это метод классификационного анализа, основные функции которого заключаются в разбиении множества исследуемых объектов на однородные группы или кластеры [7]. Задача кластеризации относится к статистической обработке.

Список литературы

1. Status codes in HTTP [Электронный ресурс]. URL: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.
2. *Bron C., Kerbosh J.* (1973), Algorithm 457 – Finding all cliques of an udirected graph, Comm. Of ACM, 16, p. 575 – 577.

3. *Роберт Седжвик.* Алгоритмы на графах = Graph algorithms. – 3-е изд. – Россия, Санкт-Петербург: «ДиаСофтЮП», 2002. – С 496.
4. *Pant G., Srinivason P., Menczer F.* Crawling the Web // In Web Dynamics / M. Levene and A. Poulovassilis, eds. Springer, 2004. P. 153-178.
5. Всё о Google PageRank [Электронный ресурс]. URL: http://designformasters.info/posts/google-page-rank/.
6. Гиперссылка [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Гиперссылка.
7. *Мандель И. Д.* Кластерный анализ. – М.: Финансы и статистика, 1988. – 176 с.
8. Jsoup Java HTML Parser 1.10.2 API [Электронный ресурс]. URL: https://jsoup.org/apidocs/org/jsoup/nodes/Document.html.
9. Intellij IDEA the Java IDE – JetBrains [Электронный ресурс]. URL: https://www.jetbrains.com/idea/.
10. JGraph mxgraph [Электронный ресурс]. URL: https://github.com/jgraph/mxgraph.
11. Javenue.csv – Java csv reader [Электронный ресурс]. URL: http://www.javenue.info/post/78.
12. HTTP - HyperText Transfer Protocol [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/HTTP.

Приложение

Код программы-краулер, основные методы.

Метод search() для обновления собранной информации.

```
    public void search(String url, String domain, int setLayer) {
        try {
            Csv.Writer writer1 = new
Csv.Writer("D:\\csvV\\"+domain+"1.csv").delimiter(';');
            Csv.Writer writer2 = new
Csv.Writer("D:\\csvV\\"+domain+"2.csv").delimiter(';');
            Csv.Writer writer3 = new
Csv.Writer("D:\\csvV\\"+domain+"3.csv").delimiter(';');
            int layer = 0;
            while (true) {
```

```
                String currentUrl;
                SpiderLeg leg = new SpiderLeg();
                if (this.pagesToVisit.isEmpty()) {
                    currentUrl = url;
                    this.pagesVisited.add(url);
                } else {
                    currentUrl = this.nextUrl();
                }
                if(currentUrl.contains("://")){
                    if(!currentUrl.startsWith("http://") &&
!currentUrl.startsWith("https://")){
                        continue;
                    }
                }
                if(currentUrl.startsWith("http://www.")){
                    currentUrl =
currentUrl.replaceFirst("http://www.","http://");
                }
                if(currentUrl.startsWith("https://www.")){
                    currentUrl =
currentUrl.replaceFirst("https://www.","https://");
                }
                URL currURL = new URL(currentUrl);
                String dName = currURL.getHost();
                if(dName.isEmpty()) {continue;}
                if(!this.pagesFoundedName.contains(dName) &&
dName.contains(domain)){
                    boolean success = leg.crawl("http://"+dName);
                    if(leg.getMessage().equals(dName)){
                        if(currentUrl.startsWith("http://")){
                            currentUrl = currentUrl.replaceFirst("http://",
"http://www.");
                        }
                        if(currentUrl.startsWith("https://")){
                            currentUrl = currentUrl.replaceFirst("https://",
"https://www.");
                        }
                        leg.crawl(currentUrl);
                        leg.links(currentUrl,domain);
                        List<String> pagesName = new LinkedList<String>();
                        pagesName.add(dName);
                        String title = leg.getTitle();
                        pagesName.add(title);

                        this.pagesFoundedName.add(dName);
                        this.pagesFounded.add(pagesName);
                        leg = new SpiderLeg();
                    }
                    if(success) {
                        leg.links(currentUrl,domain);
                        List<String> pagesName = new LinkedList<String>();
                        pagesName.add(dName);
                        String title = leg.getTitle();
                        pagesName.add(title);

                        this.pagesFoundedName.add(dName);
                        this.pagesFounded.add(pagesName);
                        leg = new SpiderLeg();
                    }
                }
                boolean succes = leg.crawl(currentUrl);
                if(succes) {
                    leg.links(currentUrl, domain);
```

```
                }
                for(int i=0;i<leg.getUrlToUrl().size();i++) {
                    if (!this.urlToUrl.contains(leg.getUrlToUrl().get(i))) {
                        this.urlToUrl.add(leg.getUrlToUrl().get(i));
                    }
                }
                System.out.println(this.pagesFounded.size() + " size of
pagesFounded");
                System.out.println(this.pagesFounded);
                System.out.println(this.urlToUrl.size() + " size of
urlToUrl");
                for(int i=0;i<leg.getLinks().size();i++) {
                    if(!this.pagesToVisit.contains(leg.getLinks().get(i)) &&
!this.pagesVisited.contains(leg.getLinks().get(i))) {
                        if(leg.getLinks().get(i).endsWith("/") &&
this.pagesVisited.contains(leg.getLinks().get(i).substring(0,leg.getLinks().g
et(i).length()-1))){continue;}
                        if(layer == setLayer-1){continue;}
                        this.pagesToVisit.add(leg.getLinks().get(i));
                    }
                }
                if(this.pagesToVisitThisLayer.isEmpty()){
                    layer++;
                    if(layer == setLayer){break;}
                    this.pagesToVisitThisLayer.addAll(this.pagesToVisit);
                }
                System.out.println(pagesToVisitThisLayer.size());
                System.out.println(pagesToVisit.size());
                if (!succes && this.pagesToVisit.isEmpty()) {
                    break;
                }
                if (this.pagesToVisit.isEmpty()) {
                    break;
                }

            }
            System.out.println(this.pagesVisited.size() + " pages visited");
            System.out.println(this.pagesFounded);
            for(int i=0; i<this.pagesFounded.size();i++){

writer1.value(this.pagesFounded.get(i).get(0)).value(this.pagesFounded.get(i)
.get(1)).newLine();
            }
            for(int i=0; i<this.urlToUrl.size();i++){

writer2.value(this.urlToUrl.get(i).get(0)).value(this.urlToUrl.get(i).get(1))
.newLine();
            }
            writer1.close();
            writer2.close();
            for(int i=0;i<this.pagesFounded.size();i++){
                for(int j=0;j<this.pagesFounded.size();j++){
                    if(j==i){continue;}
                    int count = 0;
                    List<String> temp = new ArrayList<String>();
                    for(int w=0;w<this.urlToUrl.size();w++){
                        URL url1 = new URL(this.urlToUrl.get(w).get(0));
                        String domain1 = url1.getHost();
                        URL url2 = new URL(this.urlToUrl.get(w).get(1));
                        String domain2 = url2.getHost();
                        if((domain1.equals(this.pagesFounded.get(i).get(0))
|| domain1.equals("www."+this.pagesFounded.get(i).get(0)))
                                &&
```

```
(domain2.equals(this.pagesFounded.get(j).get(0)) ||
domain2.equals("www."+this.pagesFounded.get(j).get(0)))){
                            count++;
                        }
                    }
                    if(count==0){continue;}
                    temp.add(this.pagesFounded.get(i).get(0));
                    temp.add(this.pagesFounded.get(j).get(0));
                    temp.add(String.valueOf(count));
                    this.dNameToDName.add(temp);

writer3.value(temp.get(0)).value(temp.get(1)).value(temp.get(2)).newLine();
                }
            }
            writer3.close();
        }
        catch(Exception ioe){
            System.out.println(ioe.getMessage());
        }

    }
```

Метод nextUrl() для получения URL страницы на которой будет произведено сканирование.

```
    private String nextUrl() {
        String nextUrl;
        nextUrl = this.pagesToVisitThisLayer.remove(0);
        this.pagesToVisit.remove(0);
        System.out.println(nextUrl + " next page to visit");
        this.pagesVisited.add(nextUrl);
        return nextUrl;
    }
}
```

Метод crawl() для получения ответа от веб-сервера.

```
    public boolean crawl(String url) {
        try {
            Connection connection = Jsoup.connect(url);
            htmlDocument = connection.get();
            if (connection.response().statusCode() == 200) {
                System.out.println("\nVisiting " + url);
            }
            if (!connection.response().contentType().contains("text/html")) {
                System.out.println("Retrieved something other than HTML");
                return false;
            }
            return true;
        }
        catch (Exception ex) {
            message = ex.getMessage();
            System.out.println(ex.getMessage());
            return false;
        }
    }
```

Метод links() для сбора информации.

```
    public void links(String url,String domain){
        String dName="";
        String temp_="";
        try {
            this.Title = this.htmlDocument.title();
```

```
            URL currUrl = new URL(url);
            dName = currUrl.getHost();
            System.out.println(dName);
        }
        catch(Exception ex){
            System.out.println(ex.getMessage());
        }
            Elements linksOnPage = htmlDocument.select("a[href]");
            System.out.println("Found (" + linksOnPage.size() + ") links");
            for (Element link : linksOnPage) {
                List<String> templink = new ArrayList<String>();
                templink.add(url);
                String temp = link.absUrl("href");
                try {
                    URL URL = new URL(temp);
                    temp_ = URL.getHost();
                }
                catch (Exception ex) {
                    System.out.println(ex.getMessage());
                    continue;
                }
                //System.out.println(temp_);
                templink.add(temp);
                if (temp_.isEmpty()) {
                    continue;
                }
                if (temp_.contains(domain)) {
                    this.links.add(temp);
                    if (temp_.equals(dName) || temp_.equals("www." + dName))
{
                        continue;
                    }
                    this.urlToUrl.add(templink);
                }
            }
    }
    public String getTitle(){return this.Title;}
    public List<String> getLinks(){return this.links;}
    public List<List<String>> getUrlToUrl() {return this.urlToUrl; }
    public String getMessage(){return this.message;}
```